Research Journal of Recent Sciences ______________________________ ISSN 2277-2502
Vol. **4(2),** 72-82, February **(2015)** *Res.J.Recent Sci.*

Comparative Analysis of Packet Filtering Algorithms with Implementation

Hediyeh Amir Jahanshahi Sistani[1], Sayyed Mehdi Poustchi Amin[2] and Haridas Acharya[3]
[1,2]Department of Computer Studies and Research, Symbiosis International University, Pune, INDIA
[3]Allana Institute of Management Science, Pune University, Pune, INDIA

Available online at: www.isca.in, www.isca.me
Received 6th January 2014, revised 16th March 2014, accepted 15th June2014

Abstract

Packet classification is an essential part of networking tools like firewalls and routers. All the packets have to be categorized systematically to facilitate broadband internet facilities, as well as a host of applications, embracing Internet gaming, Video on Demand (VoD), TV/Radio and e-businesses, which unceasingly necessitate an cutting-edge intensity of transmission bandwidth, meticulous Quality of Service (QoS) and elaborate security. This dissertation presents a fair appraisal and analysis of packet classification algorithms, BV, HiCuts and DimCut which are constructed on a decision tree structure. The assessment has been piloted on procedures based on analogous principles and design selections. Performance capacities have been obtained by assigning the applied classifiers in an identical set-up of trial environments. Our core involvement in this effort is an unbiased comparison with mutual norms and assessment cases, by affording a standardized appraisal of the three classification algorithms. This work stresses the evaluation of high performance packet classification systems, which are considered necessary to facilitate futuristic routers and switching systems to achieve combat security risks in a very good speed environment. In this study, we have elucidated our erstwhile recommended DimCut packet classification algorithm, and compared it with the HiCut and BV decision tree packet classification algorithms. The proposed improvements have been corroborated by simulated trials.

Keywords: Firewalls, rules, packet filtering, bit vector, hicut, dimcut.

Introduction

Packet classifiers are comprehensively employed for an array of network applications, many of which are concurrent to quality of service (QoS) requirements, and, consequently, in diverse network gadgets. High performance packet classification algorithms are of substantial benefits to both academics and industrialists. Large scale packet classification has become a key element of network security systems and Firewalls needing to classify packets, where the speed of decision making, to accept or reject, is of utmost significance.

The foremost task of a firewall is to scrutinize and select the network traffic in accordance with the designated security policy. Typically, the security policy and rules are encoded manually by a system administrator to stipulate an action for traffic flows, and, therefore, specify how to process the traffic. The security policy system spells out the progression of the network traffic[1].

Packet classifiers are extensively employed in IP networking where procedures customarily comprise one or more packet header fields. Each rule R consists of i components, where each component R [i] relates to a definite header field. When there is more than one field, the classifier is termed as multi field. A set of programme rules regulate the approval or rejection of packets. A packet classifier must compare header fields of each inward packet against a set of rules[2-4]. The packet header fields generally consist of the source IP address, the destination IP address, the transport protocol, the source port, and the destination port, which can be an exact prefix and range. Each filter R[i] has an allied action that governs how a packet P is handled if P matches R[i]. Filters can overlap; hence, a packet can match multiple filters, but the one with the highest priority among all the equivalent filters is selected as the best matching filter. Customarily, the filter's position in an ordered list of filters defines its priority.

Due to the intricacy of the search, packet classification is every so often a performance holdup in network framework; therefore, it has gathered considerable interest in the research diaspora. Broadly, there have been two major lines of research to tackle this problem: algorithmic and architectural. Several scholars have propounded a variety of algorithmic solutions, innovative algorithms as well as improvements for the contemporary algorithms However, before exploring new options, it is vital to comprehend the existing algorithms under uniform test conditions and a common set of benchmark standards[2,5].

In this study, we have elucidated our erstwhile recommended DimCut packet classification algorithm, and compared it with the HiCut and BV decision tree-based packet classification algorithms. The proposed improvements have been corroborated by simulated trials.
The rest of the paper is organized as follows: Section One reviews some related works on the various algorithms studied to

capture their advantages and disadvantages. Section Two explains the BV, HiCut and DimCut algorithms. Section Three deals with the implementation objectives. Section Four tabulates and examines the results of the comparative assessment of our experiments and findings. Section Five is a summary of our contributions and conclusion.

Related work: A packet classifier must correlate header fields of all incoming packets against a set of rules containing the security policies. Packet classification aims at pursuing the proper matching filter for a given packet header. In the multidimensional packet classification, trade memory is used for better speed and performance. When the number of rules increases, the result is inadequate, either towards search time or memory usage. Several researchers in industrial and academic fields have been endeavoring to resolve these problems, but packet classification problem continues to be a major challenge in network processing [6-12].

The bit vector search algorithm is stemmed on the core of filter set intersecting, as it is simpler to pair a part of filter at a time than to match up the whole filter totally. The packet header can be fragmented into substrings and matched with a subset of rules, where the intersection will provide a rule to match the whole packet header[13]. As against this, the Cross-Producting algorithm circuitously encodes the subsections of filters into indices that are applied to develop the keys to the Cross-Producting table. These algorithms are extremely swift and their data is principally established by the speed of partial header lookups. However, they can consume needless amounts of memory. To offset this, the data structure of 'Grid of Tries' algorithm is spread out into two fields, and utilizes a decision tree for packet classification on the source and destination address prefixes. The Cross-Producting, together with a caching technique, is recommended for multiple fields and larger classifiers with non-deterministic classification time[8].

Baboescu, Singh, and Varghese are researchers who have proposed Extended Grid-of-Tries (EGT) which fundamentally withstands multiple fields. It is notable that the EGT modifies the switch pointers to jump pointers that maneuver the search to all feasible matching filters, rather than only to the filters with the longest matching destination and source address prefixes[14]. Woo's modular packet classification, Multidimensional Cuttings (HyperCuts) and Hierarchical Intelligent Cuttings (HiCuts), employs filter set splitting method in algorithms, where the preprocessing of rule sets utilizes the strategy of cutting of the multi-dimensional space recursively to construct the decision tree[2,3,14].

Tuple Space Search algorithm is established on filter set grouping, where filters in a set are reorganized into distinct subsets with explicit conventional features. Corresponding lookups can be operated in each of these smaller subsets. The most appropriate match is extricated from the results of all the lookups. Lookups in each tuple can be systematized by means of a basic hash table. When each tuple is ascribed a hash table, the lookup can simply entreat all the hash tables to discover the most useful matching filter. The numbers of tuples govern the storage and look up time, and to arrive at the conflict-free filters feature one can employ the binary search with the objective of minimizing the number of hash queries. The Compressed Tuple Space Search algorithm also uses this fundamental method[15,16].

A study of the existing algorithms indicates that a single algorithm can never handle all the situations seamlessly, as every technique has its own advantages and disadvantages. Therefore, understanding the problem from a high level perspective can provide insights for additional improvements. Many researchers have examined and illustrated the problems of packet classification, and several solution algorithms have been suggested, but it still remains problematic, leaving innumerable opportunities to improve algorithm performance in the existing algorithms[1,17,18.]

Bit Vector, HiCut and DimCut Algorithms: Bit Vector: The linear understanding of packet classification divulges some basic concepts in what manner the data configurations can be created and how to characterize packet filters. In geometric view, several algorithms take on either 'cutting' or 'projection' techniques in multidimensional space to preprocess filter sets. The 'cutting technique' portions the space into smaller sections at designated vantage points. Each sub-section, therefore, encompasses a smaller number of filters. This procedure helps to contract the latitude of the search. The 'projection technique' plans the end-points of ranges to each dimensional axis. Two contiguous points outline an elementary intermission that is completely encompassed by a distinctive subset of filters. 'Projection technique' has advanced granularity than the 'cutting technique' and can, therefore, distinguish filters in a superior manner. However, locating an elementary interval by this technique is more challenging than tracing a sub-region by the 'cutting' technique. The decision tree-based algorithms typically apply the 'cutting' technique, while the decomposition-based algorithms customarily utilize the 'projection' technique.

The Bit Vector algorithm is a breakdown-based algorithm that depicts the subset of filters for each partial match by means of bit vectors. The filter set intersecting concept is that it is easier to match a partial filter, rather than the entire filter, at one time. Therefore, when the packet header is segregated into a set of substrings, then each substring can match a subset of filters. The intersection of these subsets is precisely the filters matching the total packet header[13].

BV employs a geometric view of the filter set and draws filters into d-dimensional space. The projections from the "edges" of the d-dimensional rectangles, identified by the filters, express elementary intervals on the axes. For each elementary interval on the axis, we define an N-bit bit-vector. Each bit position tallies with a filter in the filter set, and is prioritized. All bit-vectors are initialized to all ‘0’s. For each bit-vector, the bits are set analogous to the filters that overlap the allied elementary interval. For each dimension d, an independent data structure is constructed, which detects the elementary interval covering a given point, and then returns the bit-vector related to that interval. Once all the bit-vectors are computed, the d-data structures are constructed and searched with the corresponding packet fields independently to identify all d-bit vectors from the field searches. The most significant ‘1’ bit in the result symbolizes the highest priority matching filter. Multiple matches are easily reinforced by examining the most significant set of bits in the resultant bit vector. Consider the example in 2D filter set; shown in figure-1, each filter appears to be a rectangle on this 2D plane. The preprocessing step of the algorithm projects the edges of the rectangles to the corresponding axis, means project the end points of each rectangle to the axis, and any two adjacent projection points on an axis defines an elementary interval which is fully covered by a set of filters. In the example shown, the three rectangles create six intervals in each axis. In the worst case, the projection results in maximum of (2n + 1) intervals on each dimension. Then associate a bitmap with each dimension. A bit in the bitmap is set, if the corresponding rectangle overlaps with the interval that the bitmap corresponds to. Since there are 3 filters in total, each bit vector is 3-bit wide. The bit 0 is for the filter R1, the bit 1 is for the filter R2, and so on, as is shown in figure 1. To implement the BV, we used the binary search technique to build the single field lookup data structure for retrieving the bit vectors.

The Briefed BV Algorithm: i. Read rules and create an Array pointer structure, ii. For each rule: For each field : project the end points, set any two adjacent projection points as an elementary interval which is fully covered by a set of rules, Set the bit vectors for each elementary intervals which has n number of bits equal to number of rules(each bit represent a rule), iii. For each individual field of all rules construct the balanced binary search tree of elementary intervals which the leaves are the bit Vectors, iv. Use Search part: Read Packets: For each Packet, v. For each individual field traverse the corresponded Binary search tree and return retrieved Bit vector, Do bitwise AND operation on these bit vectors, Find the bits number with value1 as the rule id of the specific matched rules, Select the higher priority one as a target, Act as its action. End

HiCut: The packet classification algorithm, Hierarchical Intelligent Cuttings proposed by Gupta and McKeown. The concept of “cutting” emanates from observing the packet classification problem geometrically as shown in figure-2. HiCuts preprocesses the rule set for constructing a decision tree, with its leaves encompassing a specific number of rules. Packet header fields are used to navigate the decision tree until a leaf is reached. The rules stored in that leaf are then linearly explored for a match. HiCuts uses only four fields (dimension) to construct the decision tree. Selecting a decision principle is analogous to selecting a partitioning, or “cutting”, of the space. The algorithm uses various heuristics to select decision principles at each node that minimize the depth of the tree while monitoring the amount of memory used; more cuts at each level will result in a stouter and shorter decision tree. The number of cuts is determined by the local cutting circumstances and a global configurable space measure factor, spmf. The largest possible number of cuts is chosen, as long as the following inequality is satisfied[2]. spmf * number of rules at node r ≥ ∑ number of rules at each child of node r + number of cuts.

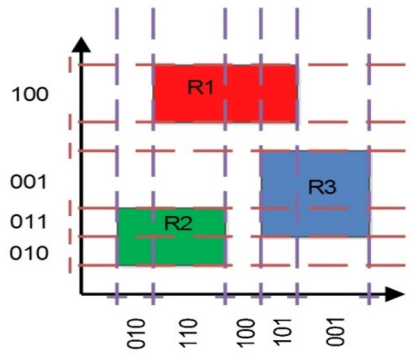

Figure-1
Shows the Projection partitiong techniqe

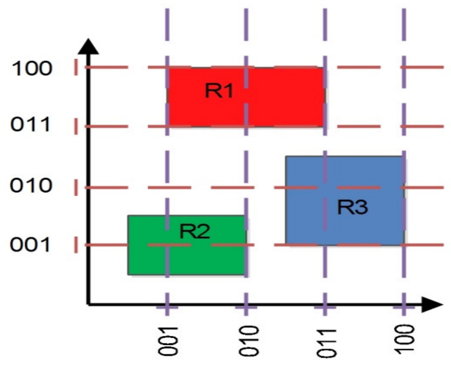

Figure-2
Shows the Cutting partitiong techniqe

The dimension to cut along each decision tree node is also critical to the algorithm performance. The algorithm gives four options. Neither one is consistently better than the others for different rule sets. The threshold is the maximum number of rules allowable in a leaf node. A higher ceiling can play a role in shrinking the size and depth of the decision tree, but will take a longer linear search time. Some enhancements, like redundancy exclusion and child node salvaging, can also be realized[7]. However, the results are not similar for diverse rule sets and options. In several situations, a few recommended algorithm options do not turn out well at all. Put into practice, the handler must delve into all routes and possibilities for ascertaining the most applicable one[12].

DimCut: The DimCut algorithm is fortified with certain alterations and enhancements on the HiCuts algorithm. DimCut is a packet classification algorithm, based on a decision tree, and using Recursive Dimensional Cutting. It has two disconnected levels, pre-processing level (tree construction) and search level. DimCut deals with the geometric view of the packet classification problem, where each rule expresses a d-dimensional rectangle in d-dimensional space, and where d is the number of fields in the rule. The algorithm pre-processes the rule set inferred to create a decision tree, and it is well expounded that the leaves contain a subset of rules with the number of rules bound by a predefined threshold. Packet header fields hunt for the proper leaf, and then linearly search for a corresponding rule fitting to that leaf[5,19].

The DimCut utilizes a heuristic for selecting the correct dimension to scan and pick the appropriate number of partitions (cut) to be made, with the objective of distributing the rules inside the partitions in a balanced manner, and with minimum repetition of feasible rules. A larger number of cuts at a node reduce the tree depth, but may increase rule replication and the number of branches, which may not achieve a good rule separation and also increase the memory usage. The process of cutting is implemented at each level, and recursively on the child nodes of that level, until the number of rules linked with each node become lower than the threshold (maximum number of rules that can be at a leaf node). We have endeavored to discover heuristics and techniques that can transform the algorithm for a higher performance with equitable memory consumption.

In DimCut, the GL (H) is the geometric length associated with column H in the rule set. To choose the best cut dimension, two fields Ha, Hb are selected which have the least GL ( ) values. Statistical regression analysis is used to estimate the best number of cuts. Based on several tests with reference to efficiency and performance , it is found that the best Number of cuts can be computed with the formula, $NC = 495.22 + (.034 * N) + (9 * 10^{-7} * N^2) + (6.23*10^{-12} *N^3)$, the Bucket size (The threshold) set as, B = 2 if N<=10000 and B=5 if 10000<N<40000 and B= 8 if 40000<=N<=100000, Here, N = Total Number of rules, some samples are provided in figures-3,4,5.

The Array Pointer structure isused which works with a large amount of rules. All rules have been arranged in priority order, in accordance with the network administrator policy. The decision tree will extend across to search the buckets covering the incoming packet and will jump to the first bucket regions of its origin. To arrive at the proper node by using the following method, it's possible to jump to the proper node rather than traversing the tree, which is the main key for the high performance and efficiency of our algorithm. The index table indexes a reference number to the proper bucket that covers the incoming packet after the optimization, such as eliminating the empty nodes, region compaction, node merging etc.

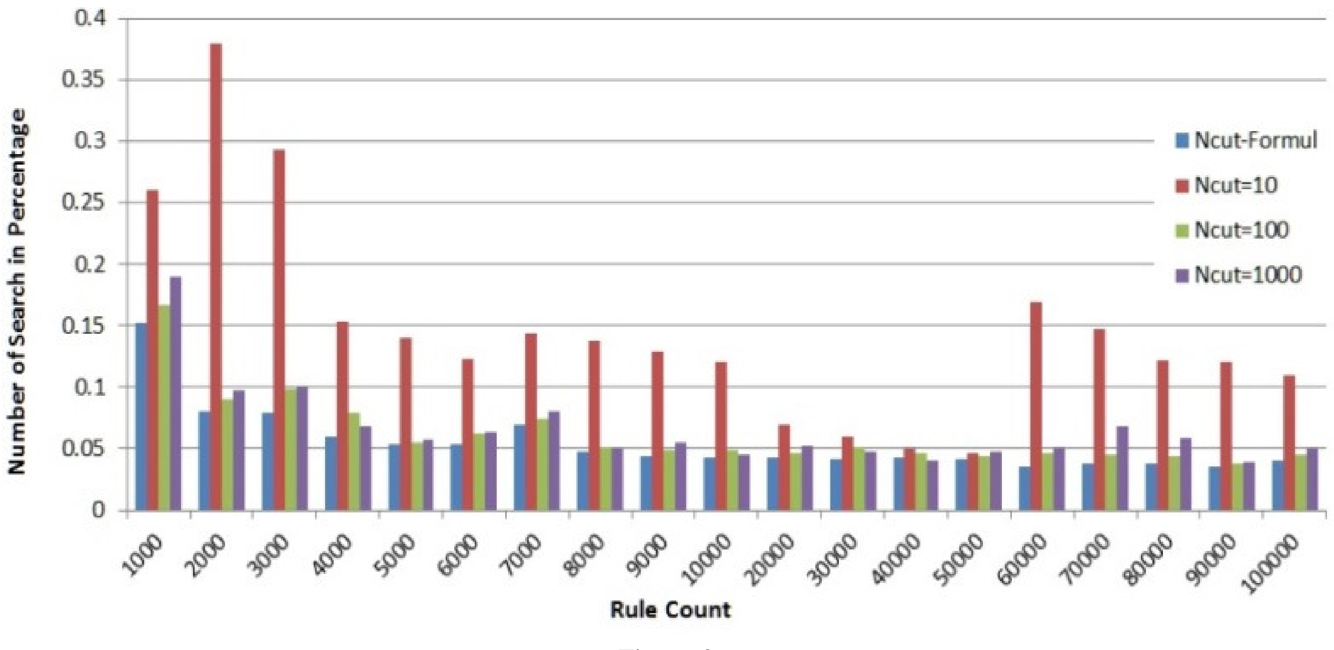

Figure-3
Test the varying number of cuts behaviour to find the better amount. In contrast, points those match with the proposed NC formula, almost shows lower number of search processing across increases in rules

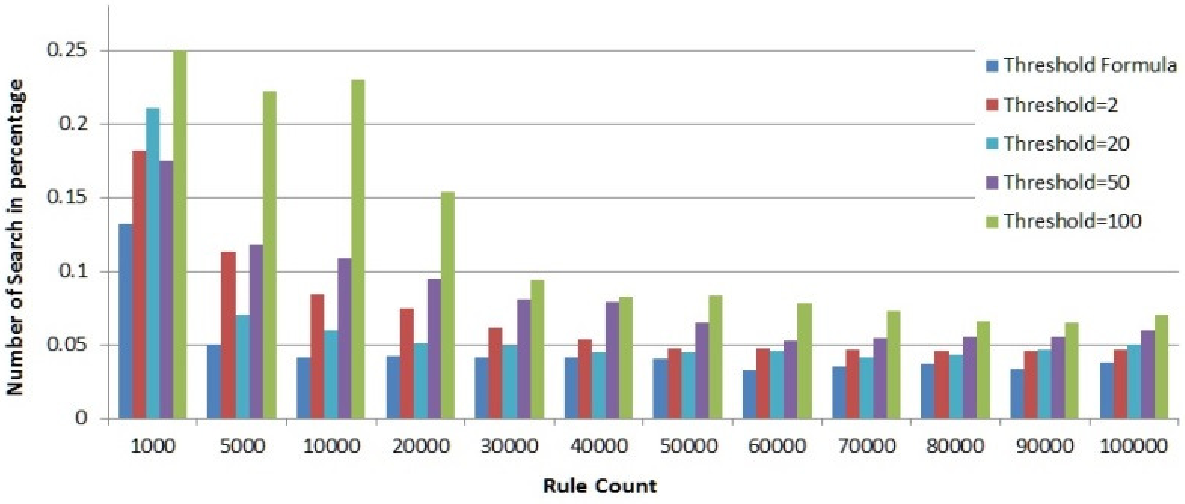

Figure-4
Test the varying Threshold (Bucket Size) behavior to find the better amount. In contrast, points those match with the proposed Threshold set, almost shows lower number of search processing across increases in rules

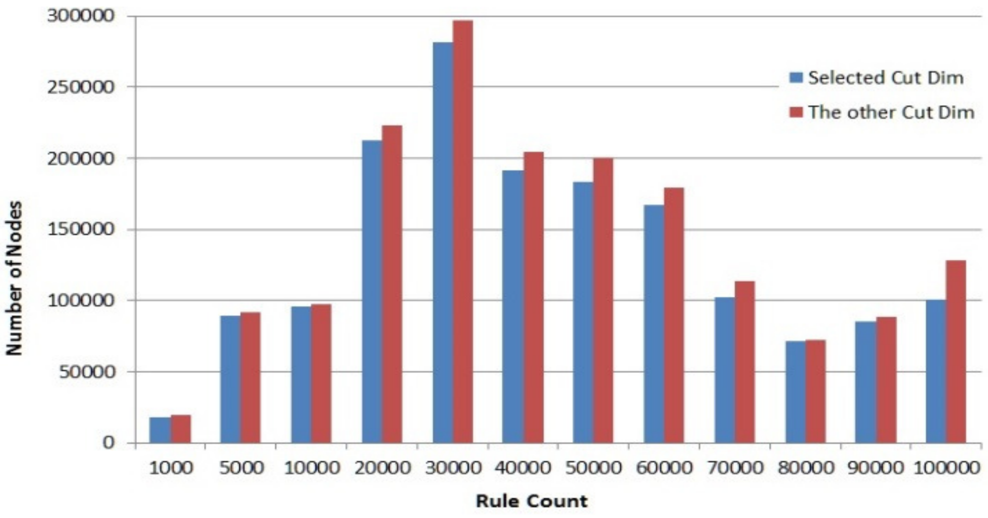

Figure-5
It Shows the Cut Dimension selection differences. In contrast, points those match with the proposed Cut Dimension selection, almost shows lower number of nodes to construct the tree across increases in rules

When the first match bucket is found, a packet will forward to all possible regions of the bucket and then all the header fields of the packet will compare to all governing rules linearly and the most prioritized rule is selected which matches perfectly, Some pseudo codes are available in figures-6,7,8,9.

Methodology

This paper presents a rational evaluation and exploration of packet classification algorithms, BV, HiCuts and DimCut which are created on a decision tree structure. The assessment has been conducted on processes built on comparable principles and design varieties. Performance measurements have been achieved by allocating the applied classifiers in an identical format of trial situations. Our basic involvement in this work is an unprejudiced comparison with shared standards and valuation circumstances, by giving a homogeneous review of these three classification algorithms which have been executed with common principles and evaluated in a common trial environment.

All the trials have been steered on standard PCs with 8 cores Intel Xeon 3.00 GHz, RAM 8.00 GB, using the Oracle VM Virtual Box to provide an insulated background, using GCC 4.7.1 compiler. Search performance is evaluated by directing it through a large number of packets and rules; and to reach the best valuation, the worst case scenario is used while providing identical settings for all tests.

```
SET G_SRC and G_DST to 0
FOR each rule in RULE ARRAY
    COMPUTE GEOMETRIC_LENGHT_SRC as subtract of broadcast and network
    source IP address
    COMPUTE GEOMETRIC_LENGHT_DST as subtract of broadcast and network
    destination IP address
    COMPUTE G_SRC as sum GEOMETRIC_LENGHT_SRC and G_SRC
    COMPUTE G_DST as sum GEOMETRIC_LENGHT_DST and G_DST
END FOR
IF Minimum of G_SRC and G_DST = G_SRC THEN
    SELECT source address as cut dimension
ELSE
    SELECT destination address as cut dimension
END IF
```

Figure-6
Find Cut Dimension – Pseudo Code

```
CREATE array BUCKET[NC] //NC=number of cut
COMPUTE r as 2^32 divided by NC
SET low to 0
FOR each bucket in BUCKET ARRAY
    SET high to SUM low and r
    SET bucket Field[cut dimension].low to low
    SET bucket Field[cut dimension].high to high
    COPY ALL rule numbers to bucket
    SET low to high
END FOR
CALL SplitBucket with BUCKET ARRAY
```

Figure-7
Initialize Buckets (nodes) – Pseudo Code

```
FOR each bucket in BUCKET ARRAY
    For each rule in bucket
        IF rule does not belong to bucket
        REMOVE rule from bucket
    END FOR
    WHILE number of rules in bucket is greater than THRESHOLD
        REVERSE cut dimension
        SPLIT bucket into number of cut
        For each rule in split bucket
            IF rule does not belong to bucket
            REMOVE rule from bucket
        END FOR
    END WHILE
END FOR
CALL OptimizeBucket with BUCKET ARRAY
```

Figure-8
Split Buckets (nodes) – Pseudo Code

```
WHILE rule exist in BUCKET
    IF packet fields match all rule fields THEN
        //Packet FOUND
        DO rule.field[action] //ACCEPT/REJECT
        RETURN
    END IF
END WHILE
// Packet NOT FOUND
DROP packet
RETURN
```

Figure-9
Search Packet – Pseudo Code

For these tests, the 1000, 5000, 10000 … 100000, numbers of random rules and the 20000 numbers of random packets have been generated, with packet size of 20 Byte and the rule size of 52 Byte.

In the figure-10, the blueprint of the simulated experiment is shown.

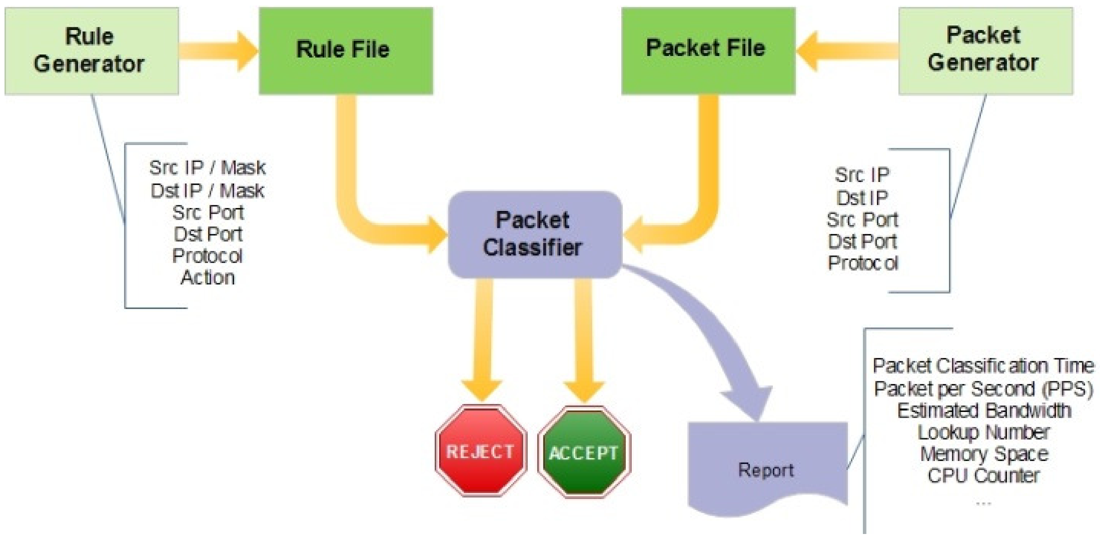

Figure-10
Simulated experimented methodology model setup

The rule's headers are Source IP (32 bit) and Destination IP (32 bit): (Exact/prefix), Source Port (16 bit) and Destination Port (16 bit): (Exact value, any, ranges), Protocol (8 bit): (TCP, UDP, ICMP, ANY, IGMP, GRE, IGP, EGP …) and the Actions (8 bit): (Accept; Deny, Log, Forward, Nothing).

The evaluation metrics and parameters institute the Cut Dimension, Number of Cut, Bucket Size, Rule Classification Time (the amount of time required to classify rules), Packet Classification Time (the amount of time needed to classify packets), Rule Memory Access, Bucket Memory Access (which indicates the amount of read or write, and the quantity of bytes to or from memory, during packet classification process among the Buckets), Number of Bytes Accessed per Packet =((Rule Memory Access + Bucket Memory Access) / Packet Count), Number of Search, Search Percentage used for evaluation and Memory consumption (the amount of maximum memory usage at the run time for both rule and packet classification),RTSC (Read Time Stamp Counter or number of CPU clock cycles ticks from the machine bootstrap).

The RTSC ("read time stamp counter") directive is accessible on processors and is a tool for accurate timing. It stores the number of elapsed clock-cycles from the moment when the processors get under way. Comparing the results of RTSC, before and after some action, could furnish the real run-time information precisely to the clock cycles.

It is significant to note that our cited applications are only for the purpose of replication and assessment, and therefore the source code is not augmented as software. We have prudently picked the formations that show the way to the best inclusive accomplishment.

Results and Discussion

The following graphs display the comparison between the HiCut, the DimCut and the Bit Vector algorithms. The incoming packets are 20000 numbers of random packets (each packet size is 20 Byte) at worst case condition. Therefore, the incoming packet doesn't match with any of the rules and offers the same fair condition for comparison of all algorithms. Figure-11 demonstrates that, while the rules are increasing the packet classification time also increases. The DimCut acts better with respect to time consumption and it is faster than the others during the progress of packet classification process.

Figure 12 shows the rule classification time, where, as the rules number increases, the rule processing ability is decreasing. However, the DimCut algorithm appears to be more competent than others. Memory usage is far higher in case of the HiCut and Bit Vector algorithms when compared to DimCut algorithm, as can be seen from figure-13. The DimCut maximum memory consumption according to this test format, for at least 100000 rules, would be near to 15 MB which is a very equitable amount. Figure 14 shows the total number of search, which explains DimCut efficiency and performance that needs to search lesser number of rules during packet classification process.

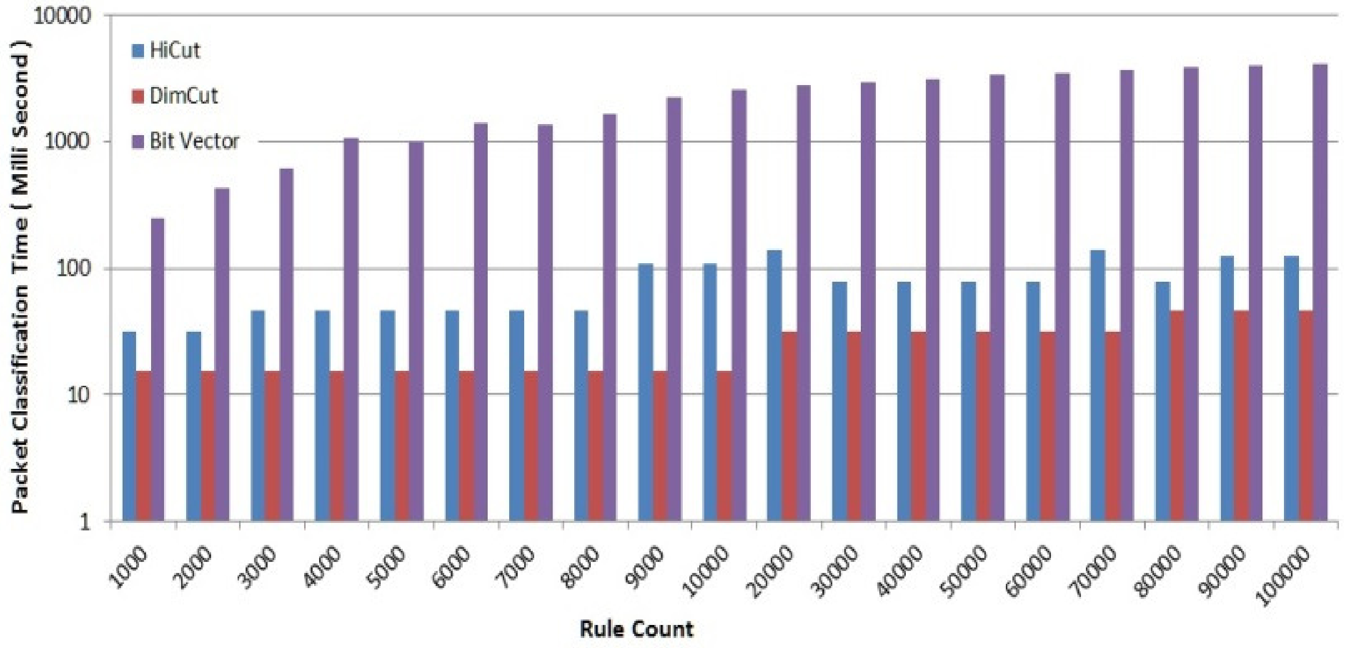

Figure-11
A comparison between the HiCut, DimCutand BitVector, to measure the packet classification time (milli second)

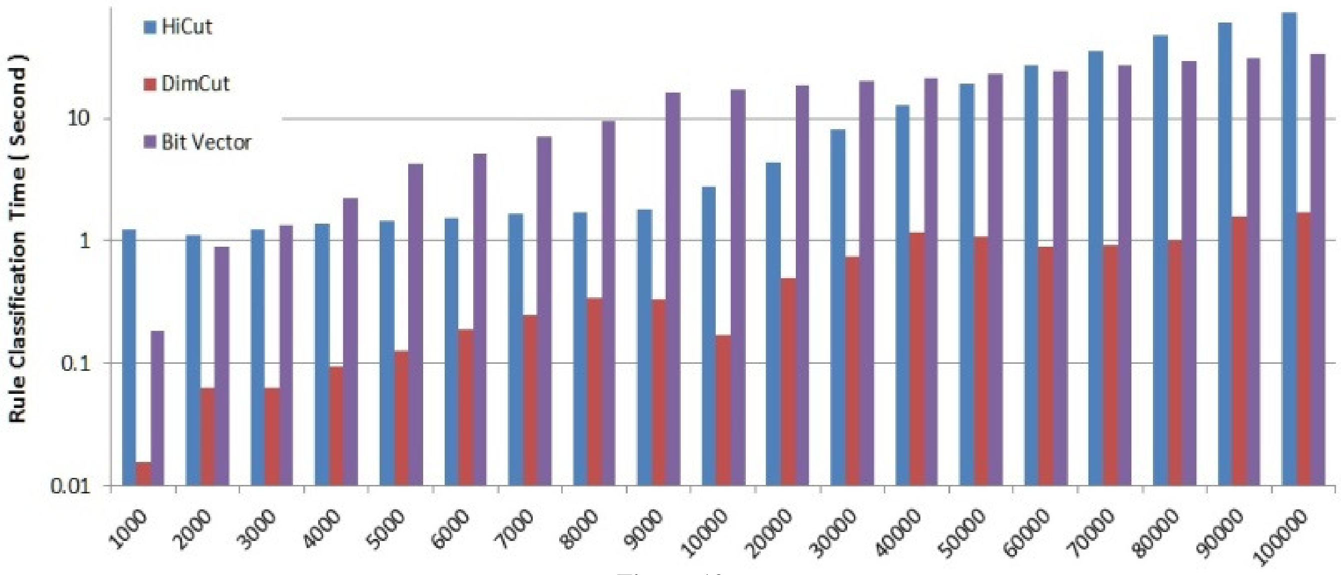

Figure-12
A comparison between the HiCut, DimCutand BitVector,to measure the rule classification time (second)

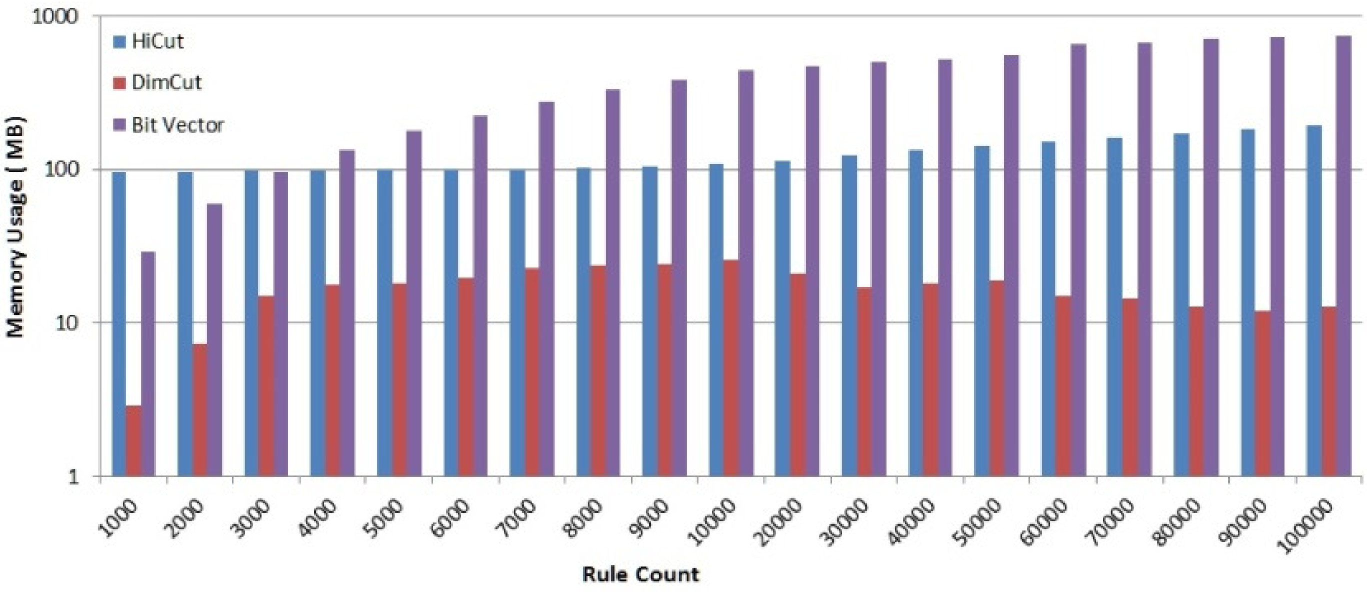

Figure-13
A comparison between the HiCut, DimCut and Bit Vector, to measure the of memory usage (M)

The number of cuts, and the dimension selection to cut at each internal decision tree node, is the key criterion for the HiCut and DimCut algorithms performance. A larger bucket size, or lesser number of cuts, can enable reduction of the size and depth of a decision tree, but it can provoke a longer linear search time. Experimenting could determine the appropriate bucket size for the best trade off of storage and throughput. Generally, a larger bucket size means a worse search processing but this does not always sustain. According to the above tests, it’s clear that the BV algorithm performance is relatively insensitive to the number of rules. Since each Bit Vector is n number of bits equal to number of rules (each bit represent a rule), and each field should make a binary search tree, a very long bit vector needs more time and memory consumption.

Through a series of experiments, we found that the algorithm reliably exhibits improved performance and accessibility with the proposed parameters and formula setting. The evaluation results are regulated in a directly analogous manner. As most packet classification algorithms are based on heuristics, different rule sets with different structures and sizes are inclined to offer very diverse results.

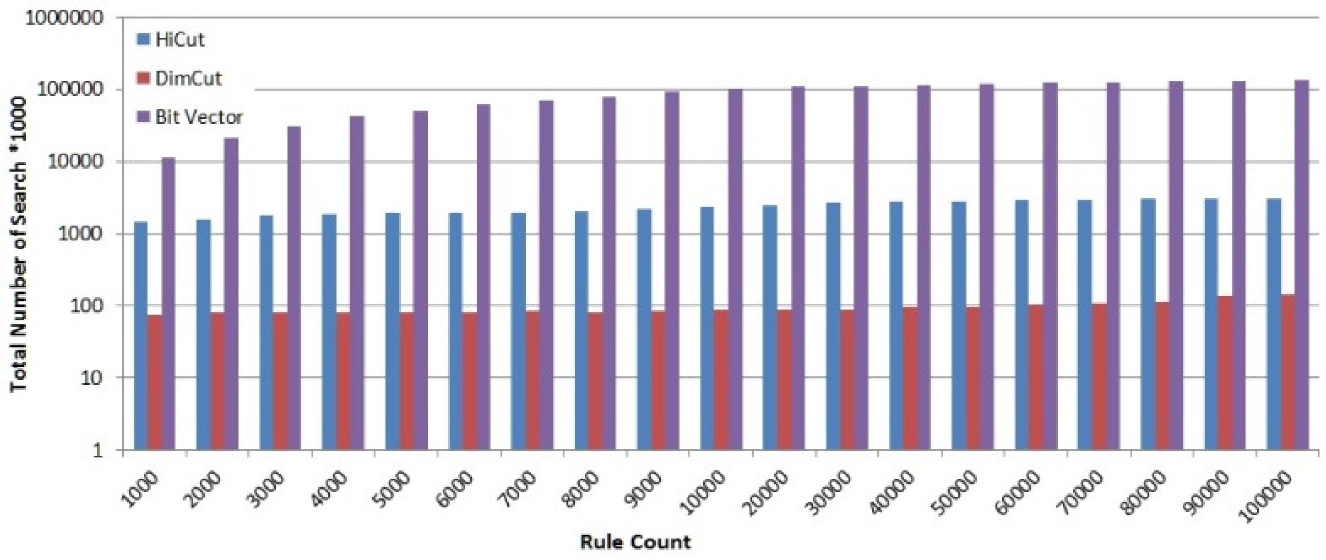

Figure-14
A comparison between the HiCut, DimCut and Bit Vector, to measure the total number of search

Table-1
The percentage of improvement of DimCut rather than HiCut, for the given Rules in worst case scenario

Rule Counts	Read Time Stamp Counter per Rule	Rule Memory Access (Bytes)	Read Time Stamp Counter per Packet
1000	99.13821855	40.09060892	84.68104645
5000	91.10609912	10.44352821	82.24620748
10000	91.38404495	5.556692323	87.45688751
50000	94.23203546	5.474968917	65.8131665
100000	97.64946636	6.638041391	96.63581364

Table-2
The percentage of improvement of DimCut rather than Bit Vector, for the given Rules in worst case scenario

Rule Counts	Read Time Stamp Counter per Rule	Rule Memory Access (Bytes)	Read Time Stamp Counter per Packet
1000	94.29717247	99.69833297	97.97863244
5000	96.91425508	99.92368711	99.186784
10000	98.98292358	99.95498707	99.5337399
50000	98.94734536	99.94240285	99.16539838
100000	99.28746719	99.94545307	99.10585489

The performance studies show that DimCut can provide an improvement of up to 96% of the given rules in Read Time Stamp Counter per Packet calculation over the HiCut and 99% over the Bit Vector, as can be seen from table-1 and 2.

The results of this test corroborate that the HiCut and Bit Vector are slower than DimCut. Both algorithms show the trend is more or less linear on the number of rules up to 10000 rules. In case of the memory usage, the HiCut's memory consumption is up to 15 times and the BV's memory consumption is up to 60 times more than the DimCut algorithm.

According to the data analysis and graphs, it is, therefore, substantiated that the proposed algorithms, based on decision tree, make packet classification faster, as compared to HiCut and BV algorithms.

Conclusion

This paper aims at the evaluation concerns for high performance packet classification algorithms, which is a vital feature in Firewalls, routers, network security and quality of service (QoS) assurance. The packet classification necessitates the packets to be unambiguously stated with the multiple packet headers, to identify the incoming flow and the rule with which the packet is to be related. It is, therefore, the central prerequisite pertaining to the range of networking management and controlling features connected with policy-based networking traffic. To accomplish a reliable performance, an algorithm should be conceived to blend the best characteristics of all approaches, besides optimizing the time-space transaction. A number of authors have put forward innovative algorithms to realize superior outcomes of classification time and memory consumption.

Our work's main contribution rests in the comprehensive and uniform appraisal of HiCut, Bit Vector and DimCut classification algorithms that have been implemented with common principles and evaluated in a common test bed, by measuring the Packet Classification Time, Number of Packet per Second Classification, Number of Search, Rule Memory access, Preprocessing Time/Tree Construction Time, Number of buckets (leaves), Depth of the tree structure and Threshold. Each test has been repeated three times to work out the average amount for the final results.

The concluding findings have been illustrated in graphics for operational demonstration. We opine that DimCut can be a viable Packet Classification algorithm that delivers a robust execution, besides allowing room for system designers to substitute components, and as a result aid the research and design community overall.

Further studies would, therefore, be necessitated to delve into more orderly systems for honing the configurable parameters, in order to upgrade the adaptive decision-tree construction procedures and rule set structure.

References

1. Sundström M., Time and Space Efficient Algorithms for Packet Classification and Forwarding, Doctoral Thesis, Luleå University of Technology Department of Computer Science and Electrical Engineering Centre for Distance Spanning Technology, **(2007)**

2. Singh S., Baboescu F., Varghese G. and Wang J., Packet Classification using Multidimensional Cutting, in Proceedings of the ACM SIGCOMM '03 Conference on Applications, Tech., Archi., and Protocols for Computer Communication (SIGCOMM '03), 213–224, **(2003)**

3. Gupta P. and McKeown N., Packet Classification Using Hierarchical Intelligent Cuttings, in Proceedings of IEEE Symp. High Performance Interconnects (HotI), 7, **(1999)**

4. Vamanan B., Voskuilen G. and Vijay kumar T.N., EffiCuts: optimizing packet classification for memory and throughput, in Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM, New Delhi, India, **(2010)**

5. Amirjahanshahi H., Poustchi M. and Acharya H., Packet Classification Algorithm Based on Geometric Tree by using Recursive Dimensional Cutting (DimCut), *in proceeding of the Research journal of Recent Sciences,* **2(8)**, 31-39 **(2013)**

6. Taylor D., Survey and Taxonomy of Packet Classification Techniques, in Proceedings of *ACM Computing Surveys*, (CSUR), **37(3)**, 238-275 **(2005)**

7. Song H. and Turner J., Toward Advocacy-Free Evaluation of Packet Classification Algorithms, in *IEEE Transactions on Computers,* **60**, **(2011)**

8. Srinivasan V., Varghese G., Suri S. and Waldvogel M., Fast and scalable layer four switching, in Proceedings of ACM Sigcomm '98, 191-202,Vancouver, Canada, **(1998)**

9. Waldvogel M., Varghese G., Turner J. and Plattner B., Scalable High Speed IP Routing Lookups, in Proceedings of the *ACM SIGCOMM,* 25-38 **(1997)**

10. Srinvasan V.and Varghese G., Fast Address Lookups Using Controlled Prefix Expansion, in Proceedings of the ACM Transactions on Computer Systems, Sigmetrics '98/Performance'98, *Joint International Conference on Measurement and Modelling of Computer Systems,* **(1999)**

11. Gupta P.and McKeown N., Packet Classification on Multiple Fields, in proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication in ACM SIGCOMM'99, 147-160 **(1999)**

12. Feldmann A. and Muthukrishnan S., Trade-offs for Packet Classification, in Proceedings of the IEEE INFOCOM, *Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies,* **3**, 1193–1202 **(2000)**

13. Stihdis D. and Lakslunan T.V., High-speed policy-based packet forwarding using efficient multi-dimensional range matching, in Proceedings of ACM Sigcomm, 203-214, Vancouver, Canada, August 31– September **(1998)**

14. Qi Y. and Li J., An efficient hybrid algorithm for multidimensional packet classification, in Proceedings of the *International Conference Communication, Network, and Information Security,* MIT, Cambridge, MA, USA, October, 9–11, **(2006)**

15. Song H., Turner J. and Dharmapurikar S., Packet Classification Using Coarse-Grained Tuple Spaces, in Proceedings of the ACM/IEEE Symp, Architecture for Networking and Comm., Systems (ANCS '06), 41- 50 **(2006)**

16. Srinivasan V., Suri S.and Varghese G., Packet Classification Using Tuple Space Search, Proc. ACM SIGCOMM, citeseer.ist.psu.edu/srinivasan99packet.html, **(1999)**

17. Baboescu F. and Varghese G., Scalable Packet Classification, ACM SIGCOMM, **(2001)**

18. Abdelghani M., Sezer S., Garcia E. and Jun M., Packet Classification Using Adaptive Rules Cutting (ARC), in Proceedings of the IEEE Telecommunications, advanced industrial conference on telecommunications/service assurance with partial and intermittent resources conference/e-learning on telecommunications workshop, **(2005)**

19. Amirjahanshahi H., Poustchi M. and Acharya H., Modification on Packet Classification Algorithm Based on Geometric Tree by using Recursive Dimensional Cutting (DimCut) with Analysis, in proceeding of the *Research journal of Recent Sciences,* **3(8)**, (issue in-press), August **(2014)**